UNIVERSIDADE DE LISBOA
Faculdade de Ciências
Departamento de Informática

Portal de Informação do Serviço de E-Mail

André Filipe Barata Matoso Fernandes Luís

Mestrado em Engenharia Informática

2008

UNIVERSIDADE DE LISBOA
Faculdade de Ciências
Departamento de Informática

Portal de Informação dum Serviço de E-Mail

André Filipe Barata Matoso Fernandes Luís

PROJECTO

Projecto orientado pela Prof. Dra. Dulce Domingos

e co-orientado por Dr. Bruno Mota

Mestrado em Engenharia Informática

2008

DECLARAÇÃO

André Filipe Barata Matoso Fernandes Luís, aluno nº 29825 da Faculdade de Ciências da Universidade de Lisboa, declara ceder os seus direitos de cópia sobre o Relatório de Projecto em Engenharia Informática, intitulado "Portal de Informação dum Serviço de E-Mail", realizado no ano lectivo de 2007/2008 à Faculdade de Ciências da Universidade de Lisboa para o efeito de arquivo e consulta nas suas bibliotecas e publicação do mesmo em formato electrónico na Internet.

FCUL, 4 de Julho de 2008

Bruno Mota, supervisor do projecto de *André Filipe Barata Matoso Fernandes Luís*, aluno da Faculdade de Ciências da Universidade de Lisboa, declara concordar com a divulgação do Relatório do Projecto em Engenharia Informática, intitulado "Portal de Informação dum Serviço de E-Mail".

Lisboa, 4 de Julho de 2008

Resumo

A qualidade de um sistema de atendimento de pedidos de suporte afecta directamente a imagem de um serviço.

Na organização onde decorreu este projecto, este serviço era assegurado através da troca de mensagens de correio electrónico entre operadores. Consequentemente, o serviço apresentava problemas de desempenho e de fiabilidade (por exemplo, alguns pedidos eram atendidos por dois operadores).

Com este projecto foi concretizado um sistema de suporte a pedidos de utilizadores, tendo por base uma plataforma open source. Esta plataforma foi adaptada e estendida de forma a responder às necessidades específicas desta organização.

Adicionalmente foi concretizado um sistema de edição colaborativo de forma a facilitar a publicação de conteúdos informativos relativos às funcionalidades e passatempos do serviço de Mail. Este sistema permitiu expandir a página inicial do serviço de forma a esta conseguir assumir a forma dum canal privilegiado de comunicação com os seus utilizadores.

Parte integrante deste projecto, foi a decisão de qual plataforma utilizar como base para ambos os sistemas. Com número bastante elevado de utilizadores, era crucial pensar em possíveis problemas de desempenho desde o início.

Salienta-se ainda que algumas das funcionalidades adicionadas à plataforma open source de atendimento de pedidos de suporte foram incorporadas na plataforma original.

Keywords Aplicações Web, Trouble tickets, Suporte, Wiki, Mail.

Summary

The quality of a trouble ticketing system directly influences the public's perception of its service.

In the organization where this project occurred, this service was carried out y exchanging email messages between helpdesk operators. Thus, the service exhibited performance problems and reliability issues (for example, some tickets were handled by two different operators).

This project consisted in building a system capable of handing trouble tickets, submitted by users, having as a starting point an open source platform. This platform was then adapted and extended so that it would meet the specific needs of the organization.

Furthermore, a system for cooperative publishing was put in place, so that it would make it easier for the whole Mail service team to publish information regarding features of their mail platform and also of recurring campaigns. This system allowed them to expand the homepage into a privileged channel of communication between the service and its users.

A major step of the project was the decision of which platforms to use both systems. With a high number of expected users, performance problems were something to pay special attention from an early stage.

It should be highlighted that some of the features added to the open source platform to handle trouble tickets, were later on returned to the community and merged into the original platform.

Keywords Web applications, Trouble tickets, Client Support, Wiki, Mail.

Conteúdo

Lista de Tabelas

Lista de Figuras

Glossário

- **Trouble Tickets** – Pedidos de suporte.
- **Wiki** – Plataforma documental que permite a edição colaborativa de conteúdos, fornecendo mecanismos avançados de mediação de conflitos e de controlo de versões.
- **Software *Open Source***– é todo o software que inclua o código fonte juntamente com o produto final. Deve permitir a redistribuição sob licenças semelhantes e permitir produtos derivados.
- **Fork** – (no contexto do movimento open source) é o nome que se dá ao acto de criar um novo projecto a partir dum outro que seja open source.
- **Front end** – máquina destina a lidar com a recolha de dados e apresentação dos mesmos ao utilizador.
- **Wikitext** – linguagem criada para facilitar a edição das páginas duma wiki, em que se pode criar ligações a outros documentos apenas colocando parênteses rectos à volta do nome da página destino, entre outras.
- **Memcached** – Sistema desenvolvido no Livejournal para fornecer um sistema de cache em que os objectos são armazenados em memória, ao contrário de outras soluções que utilizam o sistema de ficheiros ou bases de dados para tal.
- **NFS** – (network file system) é um sistema de ficheiros que permite a um cliente aceder a ficheiros num servidor, através da rede, como se de uma pasta local se tratasse.
- **Balanceador de Carga** – (load balancer) um sistema que distribui os pedidos por vários servidores de forma a aliviar a carga imposta a cada um deles.

1. Introdução

Este projecto decorreu numa empresa, doravante referida como Cliente, que contratou a instituição de acolhimento, a **PrimeIT**, para melhorar o sistema de atendimento de pedidos de suporte.

O projecto nasceu para resolver uma necessidade crescente de optimizar o processo de atendimento de pedidos dos utilizadores. Para além disso, tinha como objectivo de também criar uma plataforma de edição colaborativa de conteúdos que serviriam para melhorar a comunicação entre o serviço de Mail e os seus utilizadores. Plataforma essa que seria baseado no conceito Wiki e serviria para alimentar o sítio da Internet do serviço Mail do Cliente.

O desenvolvimento foi centrado neste serviço, mas assumiu um carácter de **projecto piloto** uma vez que havia a possibilidade deste se expandir a outros serviços da empresa.

Antes do arranque deste projecto havia uma infraestrutura montada que geria o fluxo de pedidos de suporte através de troca de mensagens de e-mail. Esta troca era feita manualmente, havendo regras que eram incutidas nos operadores do centro de atendimento. Este sistema dava azo a algumas demoras e problemas. Uma plataforma especializada no atendimento de pedidos resolveria estes problemas e daria também a possibilidade de extrair indicadores importantes para medições de qualidade e eficácia dos operadores do centro de atendimento.

Durante o desenvolvimento das funcionalidades adicionais, houve uma preocupação constante com questões de usabilidade, que serão aprofundadas mais tarde.

No próximo capítulo serão descritos em profundidade os objectivos do projecto, será dada uma revisão ao planeamento e também será abordada a metodologia de desenvolvimento utilizada.

Posteriormente, no capítulo III, será documentado o trabalho efectuado para o desenvolvimento das plataformas assim como a sua instalação e integração. Neste capítulo também será mencionado o trabalho efectuado em termos de comunicação com a comunidade de desenvolvimento do software com código aberto (*open source*) e as vantagens de tal aproximação ao desenvolvimento de software. Entrará em alguns detalhes de desenvolvimento que se mostraram vantajosos para o decorrer do projecto, como foi o caso do sistema de controlo de versões **Subversion** (svn)**.** Um factor ao qual foi dado especial atenção, o da usabilidade do sistema, terá destaque numa das secções.

No capítulo III inclui também uma visão detalhada das vantagens de utilização de software de publicação baseado no conceito de Wiki e também as técnicas utilizadas para melhorar o desempenho do sistema. Dado ao volume de utilização do serviço, este foi um factor determinante na decisão relativa à arquitectura a utilizar para a Wiki.

Termina com uma análise crítica ao trabalho efectuado e análise a possível trabalho futuro.

2. Objectivos e Percurso

Este projecto insere-se num regime de *outsourcing*, em que todo o desenvolvimento é feito nas instalações do Cliente da empresa acolhedora.

O Cliente tem uma tradição longa no apoio à comunidade Open Source, logo havia uma grande preocupação em arranjar soluções que se enquadrassem com esta tradição.

Não só era importante que fossem encontradas soluções com estas características mas também, todos os desenvolvimentos adicionais que fossem feitos para complementar a plataforma escolhida, seriam, caso fizesse sentido, devolvidos à comunidade.

Houve uma constante comunicação com os coordenadores do projecto da plataforma de atendimento de pedidos de suporte. Foram várias as contribuições que acabámos por fazer para o projecto, contribuições essas que acabaram por ser aceites e implementadas na distribuição do produto.

2.1 Processo de desenvolvimento

Antes do arranque do estágio já haviam sido levantados uma série de requisitos funcionais para a aplicação de atendimento de pedidos de suporte.

Como iríamos partir duma plataforma já existente, **adicionando** as funcionalidades que **faltassem**, fazia sentido utilizar um processo que aproveitasse esse facto e que conforme se fosse desenvolvendo novas funcionalidades, se tivesse a oportunidade de sujeitar a plataforma a testes reais para identificar problemas mais cedo, permitindo uma re-adaptação da orientação do projecto.

Assim sendo, o processo **iterativo** mostrou-se ideal uma vez que no fim de cada iteração tínhamos uma plataforma funcional e capaz de ser testada em

ambientes próximos do real, recolhendo assim imensas informações que permitiam avaliar quais as metas para a próxima iteração.

No fundo, em vez de se documentar todos os requisitos numa fase inicial, o desenvolvimento **iterativo** promove o desenvolvimento de funcionalidades por iterações, compostas pelas fases normais do **cascata**. Em cada iteração era feito a análise, desenho, implementação e testes. No final, obtinha-se assim um produto funcional que servia para fornecer à equipa de suporte que utilizaria a plataforma para gerir pedidos internos, podendo assim habituar-se ao novo sistema assim como fornecer preciosas informações que iriam ser extremamente úteis na iteração seguinte.

A equipa ligada a este projecto era muito reduzida, logo a necessidade de criar volumosos documentos era menor do que noutros projectos. Escolheu-se este modelo pois apesar do projecto ser focado no serviço do **Mail**, o departamento que iria utilizar a fundo o sistema era o de **Suporte ao Cliente**. Durante o decorrer do projecto sugiram algumas sugestões que não pareciam óbvias no arranque mas com a utilização no dia-a-dia, as necessidades foram-se revelando.

2.2 Planeamento

Apesar de utilizarmos um processo com pouca capacidade para prever a duração exacta do projecto, efectuámos um planeamento temporal das tarefas a serem desempenhadas.

As tarefas a desempenhar ficaram planeadas da seguinte forma:

Outubro 2007

- Escolher plataformas (1-15 Outubro)
    - Base de Conhecimento
    - Atendimento de Pedidos de Suporte

Outubro a Fevereiro 2007

- Desenvolver funcionalidades adicionais (15 Outubro a 29 Fevereiro)

- Atendimento de Pedidos de Suporte
    - Capacidade de Atribuir pedidos a Operadores (Out)
    - Permitir invocação de Webservices para efectuar operações de resolução de problemas (Out - Nov)
    - Notificações com configurações versáteis (Nov - Dez)
    - Histórico de Eventos por pedido (Dez)
    - Notas Privadas com pesquisa (Dez - Jan)
    - Respostas Pré-definidas (Fev)
    - Melhorias de Usabilidade (Fev)

- Base de Conhecimento
    - Preparar arquitectura para suportar adaptação visual (temas). (Out)
    - Preparar arquitectura para vários servidores (Dez)
- Testes (Fev)

Março a Abril de 2008

- Afinação da plataforma para ambiente multi-servidor (Mar - Abr)
- Integração do Design (Mar)
- Dar formação à Equipa de Colaboradores (Abr)

Abril e Junho 2008

- Testar a plataforma
    - Utilizar filas com baixa afluência para testar a plataforma de Atendimento de Pedidos de Suporte (Abr - Jun)
- Realização do Relatório Final (Abr-Jun)

Para além destas tarefas, há algumas que decorreram durante **todo o projecto**. São elas:

Outubro 2007 - Junho de 2007

- Reportar aos coordenadores do projecto OpenSource
    - Descrever alterações realizadas

- Fornecer relatórios pormenorizados das implementações das funcionalidades relevantes ao projecto

- Manter plataforma actualizada com versão actual.

O processo utilizado possibilitou-nos ainda identificar necessidades adicionais durante o projecto que não haviam sido descritas no arranque. Especialmente durante a fase de testes da plataforma utilizando uma fila – categoria de pedidos – interna com baixa afluência. Identificámos nuances, maioritariamente pormenores de interface, que fomos limando ao longo do projecto.

Durante este período, o Cliente iniciou a sua internacionalização. Lançou serviços, incluindo o **Mail**, para o país de **Cabo Verde**. Nessa altura, foi necessário adaptar as plataformas utilizadas para permitir vários domínios, uma vez que a expansão já está planeada para outros países Africanos.

Nos pedidos de suporte, foi necessário adaptar tanto a interface como os mecanismos de resposta aos pedidos. Na wiki, bastou suportar temas diferentes para cada domínio e alojar as páginas de cada país em categorias (*namespaces*) separadas.

Numa análise *post-mortem*, o planeamento foi seguido na sua maioria. Apenas houve um atraso no cumprimento do lançamento da Wiki, que se deveu a atrasos burocráticos no que toca a equipamento para suportar a mesma, assim como obrigação de respeitar os calendários de ciclos de lançamento de novas versões do serviço.

3. Componentes do Sistema

Este projecto envolvia dois grandes componentes. A plataforma de atendimento de **Pedidos de Suporte**, que existiria como um serviço individual sem estar directamente anexado a um outro serviço (por exemplo, o Mail). Desta forma poderia-se passar a fazer o atendimento dos pedidos de outros serviços nesta plataforma sem quaisquer problemas.

A plataforma de documentação, estaria sob a forma de site do serviço de Mail. Estes dois componentes são autónomos entre si, no entanto, os utilizadores do Mail seriam direccionados para a plataforma de Suporte no caso da informação disponível não ser suficiente para resolver os seus problemas.

Neste capítulo vamos analisar os sub-componentes de cada um deles, assim como as ferramentas utilizadas para os construir e a escolha das plataformas utilizadas.

3.1 Escolha das Plataformas

Parte decisiva do projecto foi a da escolha das plataformas a usar. A vontade de utilizar uma plataforma opensource, partia não só duma questão de filosofia mas também dum ponto de vista pragmático. Partido duma solução já testada pela sua comunidade pouparia algum tempo em termos não só de desenvolvimento mas também de testes.

Foi assim que, recorrendo a sites de referência na área do software livre como é o caso do SourceForge [1], e FreshMeat [2] conseguimos testar várias plataformas de pedidos de suporte.

Para tomar a decisão final, foi levado em linha de conta factores que incluem o domínio da linguagem de programação PHP (Hypertext Pre-processor) no cliente, por questões de manutenção da plataforma no futuro procurava-se uma solução baseada nesta linguagem. Procurava-se também uma plataforma que fosse simples mas ao mesmo tempo robusta. Após termos

testado várias soluções, entre elas o **OTRS** [3] que não só se revelou complexo demais, como também é desenvolvido em Perl.

Considerámos então o **osTicket** [4], após termos encontrado algum feedback sobre este sistema de trouble tickets nos site já referidos. No entanto, os responsáveis pelo projecto tinham removido quaisquer instalações do seu site, mantendo apenas uma promessa vã dum lançamento duma nova versão, completamente reescrita, algures num futuro próximo. Só este factor, levou-nos a desistir e continuar à procura uma vez que era necessário uma plataforma robusta para lidar com as inúmeras subtilezas dos variadíssimos clientes de Mail utilizados hoje em dia.

Foi então que surgiu o **eTicket** [5] que nasceu das cinzas do osTicket e foi, durante os últimos anos, angariando uma comunidade à volta da plataforma que através do *feedback* fornecido foi construindo uma plataforma robusta não só em termos de funcionalidades mas também de tratamento de emails.

Quanto à base de documentação, recorreu-se a um site independente, **WikiMatrix** [6], que permite analisar as várias plataformas wiki disponíveis no mercado apresentando uma matriz de todas elas, comparando as suas funcionalidades. Através desta informação foi possível chegar à conclusão que a mais completa era também a plataforma de Wiki mais compatível com as necessidades do projecto. Desde a extensibilidade através de plugins, assim como mecanismos orientados à melhoria do desempenho, foram argumentos de peso. Foi assim que optámos pela **MediaWiki** [7], plataforma na qual assenta a Wikipedia [8].

Havia a necessidade duma plataforma que permitisse instalação em ambiente multi-servidor, e, como já foi referido, que o fizesse com um bom desempenho. Para isto ser possível, era desejável que se utilizasse um sistema de armazenamento temporário (*cache*) de objectos em memória. Isto era crucial, pois era imperativo que se poupasse os recursos das máquinas assim como mantendo os tempos de resposta os mais baixos possível.

A **Mediawiki** suporta um destes sistemas, que também já era utilizado pela plataforma de Webmail, o **Memcached** [9].

Estavam, assim, feitas as escolhas.

3.2 Sistema de Atendimento de Pedidos de Suporte

Como já foi descrito anteriormente, a plataforma escolhida foi o **eTicket**. Este, possui capacidades de leitura de mail via **POP3** (Post Office Protocol), que se trata do protocolo de obtenção de mail mais comum. Permite também configurar o servidor de mail através do chamado **Pipe**, que não se trata nada mais do que definir um script em **Perl** que irá ser executado na recepção duma mensagem (o conteúdo desta mensagem é passado ao script como sendo Standard Input).

Uma vez que o Cliente já possuía uma plataforma de apoio à **criação de pedidos de suporte**, que ajuda a eliminar dúvidas no processo de recolha de dados, interligou-se esse sistema à plataforma através do envio de e-mails. Portanto todos os pedidos criados por utilizadores, chegariam à plataforma via email. Os únicos a serem criados através do eticket seriam os dos Operadores do Call Center, como se pode ver na **Figura 1**.

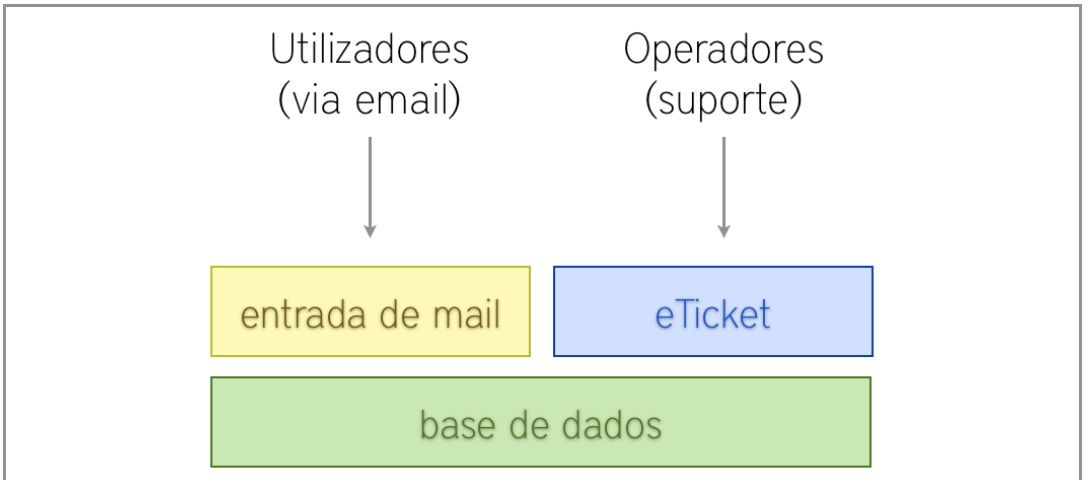

Figura 1 – Esquema de criação de pedidos de suporte

3.2.1 Funcionalidades Adicionais

Apesar da plataforma estar munida dum conjunto de funcionalidades satisfatórios no que toca a gestão de pedidos, havia algumas lacunas. Não

permitia associar pedidos a operadores, nem tão pouco manter um histórico dos eventos decorridos ao longo da vida dum pedido. Era, portanto, necessário fazer um levantamento de requisitos. Foi então que compilámos a seguinte lista de desenvolvimentos a serem feitos sobre a base fornecida pela plataforma escolhida.

- **Associar pedidos a operadores** – isto para que não ocorresse a situação em que um pedido está a ser atendido por vários operadores.

- **Histórico de Pedidos** – Guardar um registo cronológico de todos os eventos relevantes a cada pedido.

- **Notificações** – Capacidade de configurar notificações para os utilizadores, operadores associados ao pedido e membros duma determinada fila de pedidos.

- **Notas Privadas** – Os operadores poderão adicionar notas privadas a cada pedido, sendo posteriormente pesquisáveis a partir do motor de pesquisa.

- **Acções (webservices)** – Permitir a extensão *a posteriori* com classes que implementem invocações de webservices remotos.

Para além destas novas funcionalidades, haviam alguns elementos chave que apesar de já existirem na plataforma, teriam de ser **melhorados** dada a sua importância.

Desde cedo que nos apercebemos que o **ponto de entrada** de E-Mails requeria uma robustez enorme, uma vez que o panorama de aplicações cliente de correio electrónico nos dias de hoje é assustadoramente variável.

Existem duas formas de codificar texto: **quoted-printable** e **base64**. Mapas de caracteres (*charsets*) existem inúmeros: **utf-8**, **utf-16**, **iso-8859-1** (latin1), **windows-1252**, etc. Existem ainda variadas formas de enviar várias partes (vulgos anexos e versões alternativas da mensagem) no mesmo email: **mixed**, **message**, **digest**, **alternative**, **related**, **signed** e **encrypted**.

Todas estas combinações, provocam imensos problemas ao receber uma mensagem. Dada a natureza das mensagens, é imperativo que não se percam mensagens uma vez que quando se trata de suporte ao cliente, pode bastar perder **uma** mensagem para afectar uma **grande** parte da imagem do produto perante um (ou mais) dos seus utilizadores.

Assim sendo, uma das mais **importantes** funcionalidades adicionais, foi o **suporte** a todos os diferentes **tipos** de mensagens de correio-electrónico. Funcionalidade que, desde logo, foi devolvida ao projecto original, o eTicket.

3.2.2 Extensão do Modelo de Dados

Para implementar as novas funcionalidades foi necessário criar novas entidades no modelo de dados. Na **figura 2**, na próxima página, está o novo modelo de dados, em que as entidades existentes na plataforma original estão marcadas com um asterisco.

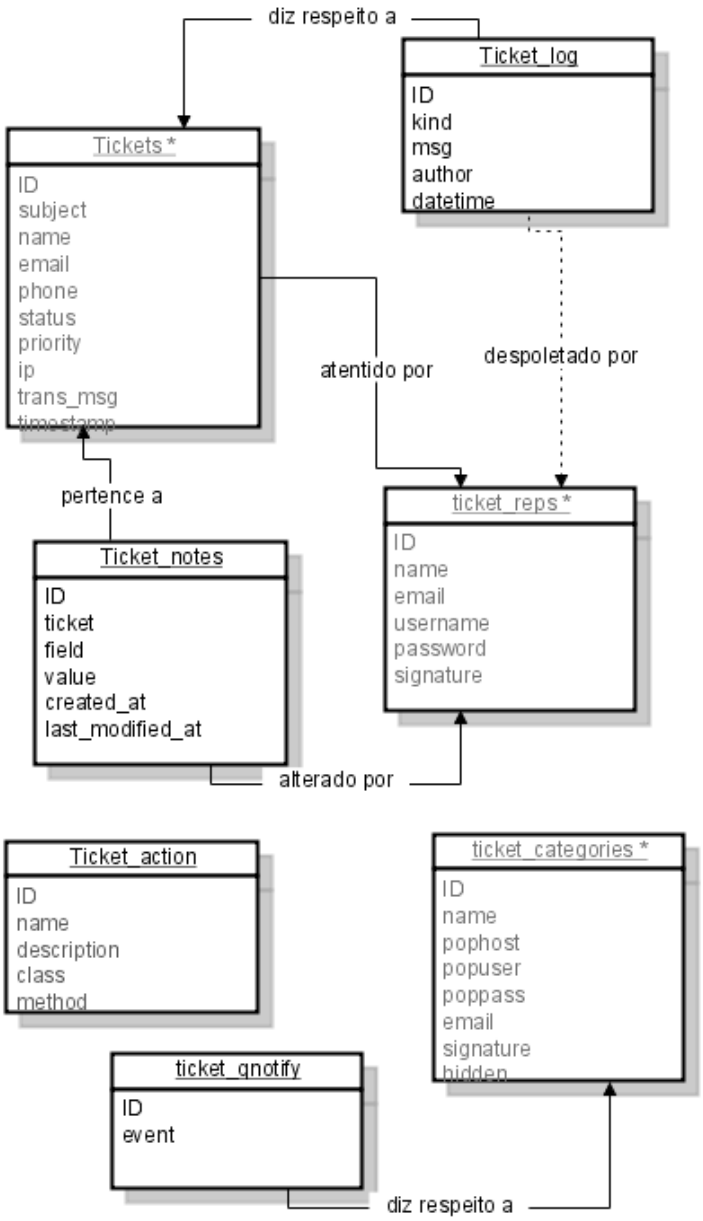

Figura 2 - Entidades adicionadas ao modelo de dados.

Relações entre as entidades e as funcionalidades criadas:

- Ticket_log: usado no histórico de eventos relativos a cada pedido.
- Ticket_action: invocação de webservices (e não só) para efectuar acções sobre um determinado pedido.
- Ticket_notes: notas privadas, pesquisáveis.
- Ticket_qnotify: configuração de notificações para cada fila (categoria de pedidos).

3.2.3 OpenSource: Retorno Contínuo

Ao longo do projecto, foi necessário manter uma comunicação com a comunidade à volta de projecto de forma a manter um sentido bidireccional na transferência de conhecimento.

Toda a comunicação foi feita através do já mencionado **SourceForge**, onde o projecto reside. Eram colocadas mensagens nos fóruns para discutir novas funcionalidades, dar sugestões ou mesmo pedir ajuda sobre um dado tema relacionado com o projecto.

Uma vez que este é um projecto mantido por voluntários, sem quaisquer remunerações, houve, como seria de esperar, algumas semanas em que o projecto estagnou. Não havia movimento por parte da comunidade nem havia receptividade às nossas propostas.

Nessa altura, o custo de continuar anexado ao projecto foi medido, uma vez que apesar de podermos implementar as nossas funcionalidades, cabia aos coordenadores do projecto a decisão de integrar as nossas sugestões na plataforma distribuída por eles.

Havendo sempre a alternativa de fazer o que se chama de *fork* no projecto, isto é, passar a distribuir um novo sistema de atendimento de pedidos de suporte, sobre um novo nome mas desta vez com todos os poderes de decisão do nosso lado.

Permanecer com o eTicket	Fazer o fork – Novo Projecto
+ Actualizações continuariam após a finalização do período de desenvolvimento do projecto + Comunidade activa - Menos poder de decisão	+ Total liberdade - Perda da comunidade - Custo adicional a manter a plataforma actualizada - Para o projecto se manter vivo, seria necessário acompanhamento muito para além da duração do projecto

Tabela 1 – comparação entre possíveis panoramas para o futuro do projecto

Os custos e riscos foram medidos, e no final, percebeu-se que, dadas as circunstâncias, era mais vantajoso para o Cliente continuar no modelo até então adoptado.

Continuou-se a comunicação das funcionalidades implementadas por nós e grande parte acabou por ser aceite e integrada na plataforma final.

A ferramenta que utilizámos para controlo de versões, o **Subversion** (SVN), facilitou imenso a comunicação de alterações, visto conseguirmos obter facilmente ficheiros que documentam todas as alterações feitas para implementação de certas funcionalidades.

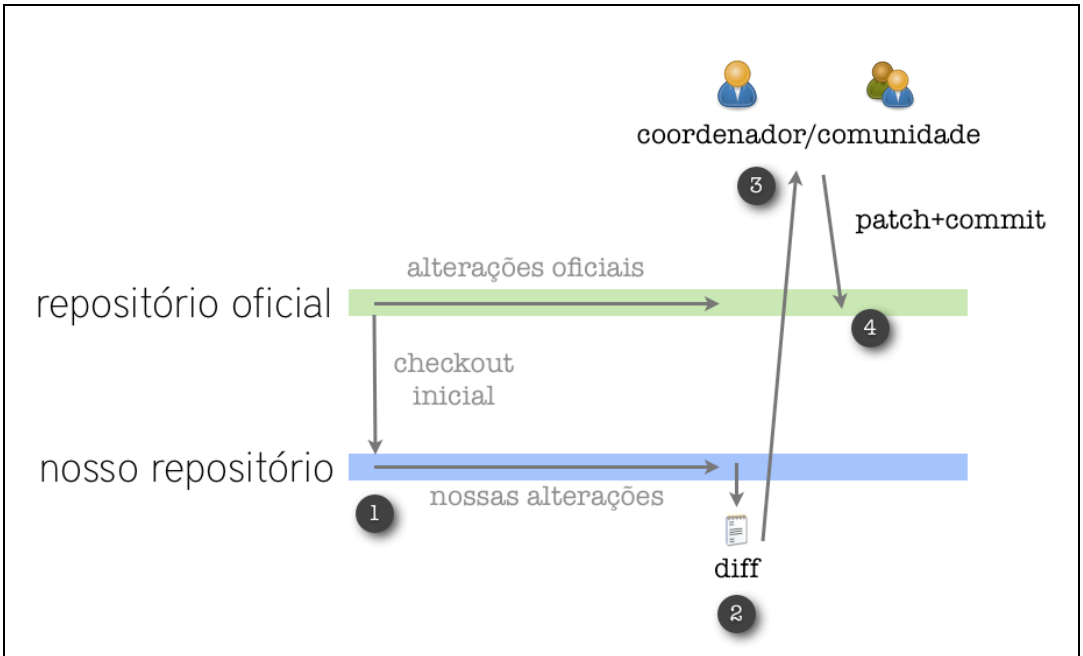

Figura 3 - Fluxo de submissão de alterações usando o SVN.

Processo descrito na figura 3:

1. Fazer *checkout* da plataforma para o nosso ambiente de desenvolvimento. (nossa base do svn);
2. Na altura de submeter as alterações para apreciação do coordenador e/ou da comunidade, obtínhamos um ficheiro com as diferenças (executando o comando `svn diff` com alguns parâmetros) contendo todas as alterações desde a base (1) e colocávamos no site do projecto para apreciação;
3. As alterações eram revistas pelo coordenador e pela comunidade;
4. Quando o coordenador achava que as alterações faziam sentido, aplicava o ficheiro das diferenças à sua cópia e guardava as alterações no repositório oficial.

Foi assim durante grande parte do projecto, até que, a certa altura, fui promovido a "**programador**" do projecto, estatuto que me permitiu efectuar alterações no repositório oficial directamente numa ou duas situações.

3.2.4 Melhoramentos na usabilidade da aplicação

Seguindo as **Oito Regras de Ouro** [10] para o Design de Interfaces de **Ben Shneiderman**, fomos colmatando algumas falhas detectadas pela nossa equipa de testes.

Mensagens informativas para fechar ciclos

Para algumas operações não eram fornecidas quaisquer mensagens de erro ou de conclusão. De facto, não é algo que possa ser dispensado pois o utilizador necessita ser informado sobre a acção que acabou de desempenhar ou, caso tenha havido erro, receber informações sobre o erro que ocorreu com o intuito de perceber se houve alguma alteração de estado.

Fornecer atalhos para utilizadores avançados

Dada a natureza do sistema, um grande número de utilizadores irá usar a plataforma diariamente. Este factor implicava uma forte preocupação em

facilitar a esses mesmos utilizadores, o acesso a funcionalidades mais usadas e sequências de acções mais frequentes.

Na verdade, este é mais um caso duma funcionalidade que foi **devolvida** ao produto original. Para facilitar a consulta de pedidos sobre diferentes categorias, foi criada uma **listagem de categorias** com o número de pedidos abertos como pode ser visto na figura 4.

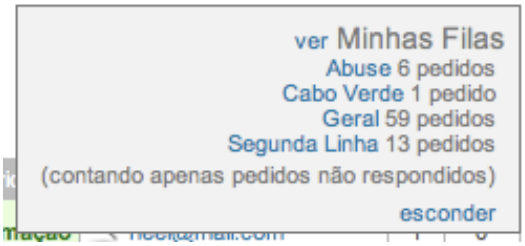

Figura 4 - Botão "Minhas Filas" expandido.

Foi implementado utilizando uma técnica chamada "**Unobtrusive Javascript**" [11] que permite que estes objectos funcionem de forma interactiva e imediata sem carregar uma nova página mas que, ao mesmo tempo, permitem uma utilização **acessível,** não só para utilizadores invisuais utilizando **leitores de ecrã,** mas também a utilizadores sem deficiências. Permite, por exemplo, que o utilizador opte por abrir o menu numa nova janela ou nova aba do browser sem que quebre a funcionalidade pretendida.

Para além desta funcionalidade, optou-se por facilitar uma **acção repetida** em **todos** os pedidos de suporte e que se pretendia que fosse a **primeira** acção desempenhada por um operador: atribuir a si mesmo o pedido para **evitar** que outro operador começasse a tratar do pedido simultâneamente. Sem este atalho visual, o operador teria de seleccionar de um dropdown o seu nome e clicar no botão "Atribuir". Desta forma não só se alerta o utilizador para a importância da acção, mas também completa a operação com apenas um clique.

Esta funcionalidade também foi devolvida ao projecto original.

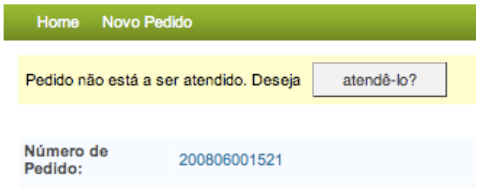

Figura 5 – Atribuir o pedido a si próprio.

Apesar de nos faltar condições para conduzir testes formais de usabilidade, o retorno que fomos recebendo da equipa de testes, indicou que as melhorias efectuadas serviram, de facto, para melhor a utilização da plataforma.

Informação de vários níveis

Na listagem dos pedidos, a equipa de testes revelou-nos que era conveniente incluir tanto o nome como o endereço de email do utilizador que submeteu o pedido. Para evitar a criação de mais uma coluna e sobrecarregar a memória a curto prazo do utilizador, optámos por um truque que ao passar por cima do endereço de email, o nome do utilizador é revelado.

Figura 6 – Informação de vários níveis.

Microformatos

Esta funcionalidade, na verdade, foi algo que surgiu devido a uma política interna de fomentar o uso de microformatos nos vários serviços. Em resumo, **microformatos** [12] são conjuntos de especificações que permitem aos autores estender o valor semântico do HTML puro. Permitem embeber dados sobre pessoas, eventos, etc. Neste âmbito, fazia sentido evidenciar os criadores dos pedidos uma vez que os operadores poderão facilmente importar os cartões de visita (vCards) dos clientes para livros de endereços individuais ou partilhados com um mero clique.

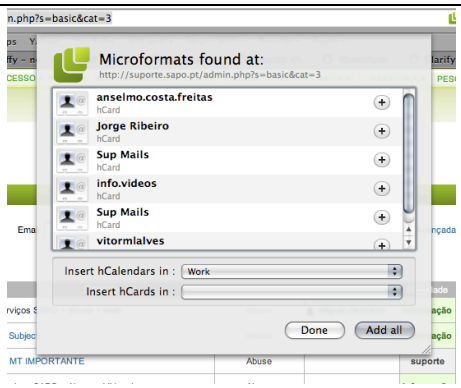

Figura 7 - Microformatos reconhecidos pelo Safari (MacOS).

Acções

Para facilitar a execução de acções que possam envolver webservices remotos foi criada uma infraestrutura que permite a invocação desses métodos a partir do ecrã de detalhes de um pedido. Uma vez anexado a um operador, esse operador terá permissões para executar essas acções. O menu é mostrado reduzido, permitindo a expansão que revela as opções.

Figura 8 - Várias fases da invocação de acções remotas.

3.2.5 Testes

Todas as funcionalidades desenvolvidas na plataforma foram testadas na sua iteração pela equipa de operadores do departamento de Suporte. Para além das entrevistas informais que conduzimos, foi sempre pedido que executassem uma série de tarefas que testassem cada aspecto da funcionalidade a ser testada. Apesar de não termos documentos sobre estes testes, foi de extrema importância uma vez que nos permitiu identificar inúmeras falhas, corrigi-las e até recolher sugestões que mais tarde viriam a ser implementadas.

Este foi o caso duma acção para efectuar o **reencaminhamento** de pedidos para endereços de correio electrónico que não o criador do pedido. Esta necessidade surgiu visto que alguns utilizadores, por lapso, enviavam problemas para este sistema de suporte que na realidade diziam respeito a um departamento completamente diferente, o de fornecimento de acesso à Internet de banda-larga.

3.2.6 Escalabilidade e Desempenho

Depois de todas as funcionalidades desenvolvidas, depará-mo-nos com alguns problemas de instalação que requeriam algum estudo. De entre elas, a mais relevante e importante, apesar de aparentemente simples, prendia-se com o formato de numeração dos pedidos de suporte.

A plataforma original fabricava os números de identificação (de seis algarismos) aleatoriamente, verificando sempre que não havia colisões com os pedidos já criados. Desta forma bastava ao utilizador fornecer o número atribuído pelo sistema e o endereço de correio electrónico utilizado para criar o pedido; com estes dois dados, o sistema permitia ao utilizar entrar no sistema e gerir todos os pedidos anteriormente submetidos.

A discussão surgiu por motivos de planeamento futuro. Com o ritmo de submissão de pedidos, um sistema aleatório com um número fixo de dígitos iria acabar por se esgotar e eventualmente criar problemas.

Começou-se a considerar alternativas.

1. 3hjb43 – combinações alfanuméricas de seis elementos;
2. A123456 – Séries incrementais numéricas marcadas com uma letra (incremental) no inicio;
3. 200806123456 – Séries incrementais numéricas, iniciadas pelo ano, mês, seguida de seis dígitos;
   (Algumas outras sugestões que foram excluídas à partida.)

Deste conjunto, conseguia-se entender algumas vantagens e desvantagens. A número 1 poderia criar alguma confusão no atendimento telefónico dos pedidos, uma vez que as pessoas revelam alguma dificuldade em memorizar combinações alfanuméricas algo complicadas. Por isso considerou-se a opção número 2, em que a única letra situava-se no início, resolvendo o problema da comunicação verbal mas limitava o número de pedidos a 23 séries de 999 999 pedidos cada uma. Não era suficiente a longo prazo. A **estimativa mensal**, baseada em números do sistema antigo, rondava os **100 000 pedidos**.

Chegou-se à conclusão que a melhor opção, ainda que mais pesada em termos de armazenamento de dados, seria mesmo a de utilizar o ano e o mês como prefixos numéricos. Desta forma, permitia-se o crescimento até perto de **um milhão** de pedidos mensais.

Esta foi uma questão que, apesar de estar presente durante todo o desenvolvimento da plataforma, só surgiu na altura de instalação da plataforma final. Algo que permitiu uma decisão cautelosa e baseada na utilização da plataforma pela equipa de **Suporte**.

Para além destas opções, tentou-se garantir que a plataforma seria capaz de lidar com um volume de pedidos considerável na sua página inicial. Para isso, utilizou-se o binário **Apache Benchmark** (ab) para fazer um teste ingénuo (estes testes dependem imenso da máquina que realiza os testes e também da ligação ao servidor).

O objectivo de desempenho era que **todos** os pedidos fossem atendidos **sem erros** e também que demorassem **menos que 1 segundo** a enviarem resposta.

Aqui fica um exemplo dos testes conduzidos.

Teste #1 – 500 pedidos, com 10 pedidos concorrentes:

```
thepassenger:~ andre$ ab -n 500 -c 10 http://XXXXXXXXXXXXXXX/
This is ApacheBench, Version 2.0.40-dev <$Revision: 1.146 $> apache-2.0
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Copyright 2006 The Apache Software Foundation, http://www.apache.org/

Benchmarking XXXXXXXXXXXXXXX (be patient)
Completed 100 requests
Completed 200 requests
Completed 300 requests
Completed 400 requests
Finished 500 requests


Server Software:        Apache
Server Hostname:        XXXXXXXXXXXXXXX
Server Port:            80

Document Path:          /
Document Length:        2656 bytes

Concurrency Level:      10
Time taken for tests:   8.712996 seconds
Complete requests:      500
Failed requests:           0
Write errors:           0
Total transferred:      1513020 bytes
HTML transferred:       1330656 bytes
Requests per second:    57.39 [#/sec] (mean)
Time per request:       174.260 [ms] (mean)
Time per request:       17.426 [ms] (mean, across all concurrent requests)
Transfer rate:          169.52 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:       10   26   8.7     26      59
Processing:    64  145  43.1    146     384
Waiting:       61  142  43.0    142     381
Total:         82  172  45.7    176     412

Percentage of the requests served within a certain time (ms)
  50%    176
  66%    192
  75%    204
  80%    210
  90%    226
  95%    236
  98%    267
  99%    290
 100%    412 (longest request)
```

O pedido mais longo demorou 412 milisegundos a ser respondido e em nenhum ocorreu um erro. (Este é um exemplo do sistema que se encontra em produção.)

3.3 Base de Conhecimento

Em termos de requisitos funcionais, a plataforma teria de permitir a edição colaborativa pelos membros da equipa do serviço de Mail mas não permitindo a edição por parte dos utilizadores em geral.

Isto vai um pouco contra o conceito de Wiki, que tipicamente funciona numa de várias formas:

- Edição permitida a todos os utilizadores registados
- Edição permitida a todos os utilizadores registados e anónimos

A forma que encontrámos de controlar os acessos, passou pela criação de contas para a equipa (5~10 pessoas) e **removendo** privilégios aos utilizadores de **criarem** novas contas.

Para o fazer, recorria-se a uma conta de Administrador, visto ser um acontecimento extremamente raro – somente aquando da contratação dum novo membro.

A plataforma que se enquadrava com todos estes aspectos era, por razões já apresentadas, a **MediaWiki**. (ver capítulo 2.1)

3.3.1 Preparação da Plataforma

Num ambiente **Wiki**, é habitual haver um controlo de versões para as páginas e objectos multimédia, uma vez que é necessário haver controle sobre as alterações efectuadas para que, caso se justifique, se possa reverter para uma versão anterior.

Na MediaWiki, todos os objectos de imagem possuem uma ligação para uma página de descrição do ficheiro com o historial de todas as versões do mesmo, assim como algumas informações adicionais, como se pode ver na figura 9.

Figura 9 – Exemplo duma página de descrição duma Imagem na Wikipédia

Numa base de conhecimento, com objectivo de informar os utilizadores, todas as imagens possuírem ligações para páginas destas, com imensas informações técnicas, não só poderia confundir utilizadores como tornam toda a experiência frustrante pois não será desta informação que os utilizadores procurarão.

Ademais, esta informação é, sim, relevante num ambiente de pura Wiki, em que todos os utilizadores que visionam as páginas poderão editá-las e actualizarem as imagens.

Desta forma, tornou-se necessário estender a plataforma para que permitisse duma forma fácil **remover esta ligação**.

Usando **WikiText**, que se trata da linguagem usada pela plataforma para elaborar as páginas, tal como o HTML, conseguimos especificar que queremos apenas a imagem.

Tipicamente insere-se imagens da seguinte forma:

```
[[Imagem:Nome_do_ficheiro.jpg]]
```

Com opções em termos de tamanho (funcionalidade crucial para a edição fácil):

```
[[Image:Nome_do_ficheiro.jpg|100px]]
```

Seguimos este padrão e adicionámos, através de desenvolvimento sobre a plataforma, a opção **raw** às imagens da Wiki.

```
[[Image:Nome_do_ficheiro.jpg|100px|raw]]
```

Desta forma todas as imagens seriam visíveis sem qualquer ligação, tal como pretendido.

3.3.2 Arquitectura da Informação

Para que toda a equipa do serviço, que a maior parte não possui conhecimentos de HTML – a linguagem usada para construir páginas, conseguisse facilmente editar a Wiki sem que esta se tornasse numa amálgama de edições avulsas, era necessário ter algum cuidado com a forma como o conteúdo era colocado.

Para isso usámos uma funcionalidade bastante útil e poderosa da mediawiki. ***Templates*** ou **predefinições**, permitem-nos criar esqueletos de pedaços de informação que irão, à partida, ser repetidas pela mesma página ou mesmo por várias.

Alguém com conhecimentos de webdesign cria esse esqueleto um única vez, enquanto que assim, os editores e colaboradores apenas terão de especificar as variáveis para serem preenchidas nos espaços do esqueleto.

Exemplo duma Predefinição (template)

```
{{Funcionalidade
    |imagem=mail5g.jpg
    |titulo=Porque agora tem mais espaço 5GB
    |descricao=Com 5GB não queremos que lhe falte espaço para comunicar.
}}
```

Produzem, sem quaisquer conhecimentos de HTML, este pedaço de informação:

Figura 10 – Apresentação dum Template/Predefinição da mediawiki.

Esta funcionalidade ajudou bastante a uniformização dos conteúdos apresentados, assim como possibilita a alteração fácil de todas as ocorrências de certo tipo de bloco bastando para isso editar a configuração da **predefinição** correspondente.

Um factor determinante no sucesso do projecto era a forma como as pessoas que fazem parte da equipa editorial do Cliente iriam receber a nova forma de inserir conteúdos.
Para isso, limitávamos ao máximo a necessidade destes explicitarem os aspectos visuais das páginas. O facto de terem um mecanismo que lhes indicava um conjunto fixo de parâmetros, limitava os desvios que naturalmente viriam a acontecer no caso de cada operador poder escolher a forma como inseriam uma funcionalidade.

3.3.3 *Deployment* em Ambiente Multi-servidor

Uma vez que a arquitectura da plataforma Web do cliente está desenhada usando vários *frontends* para suportarem o acesso de vários milhares de sessões em simultâneo, um cuidado especial em permitir a todos estas máquinas acederem ao conteúdo estático gerado pela **Wiki**.

Poder-se-ia ter implementado um sistema de replicação de conteúdos que em todas as alterações feitas por algum dos *frontends* enviaria uma cópia do ficheiro para todas as outras réplicas.

No entanto, por uma questão de simplicidade e também tendo em vista manter a ocupação em disco ao mínimo – de relembrar que alguns dos objectos podem ser multimédia – foi configurando uma pasta que será distribuída por todos os nós da rede através de *mountpoints ntfs* em cada um deles.

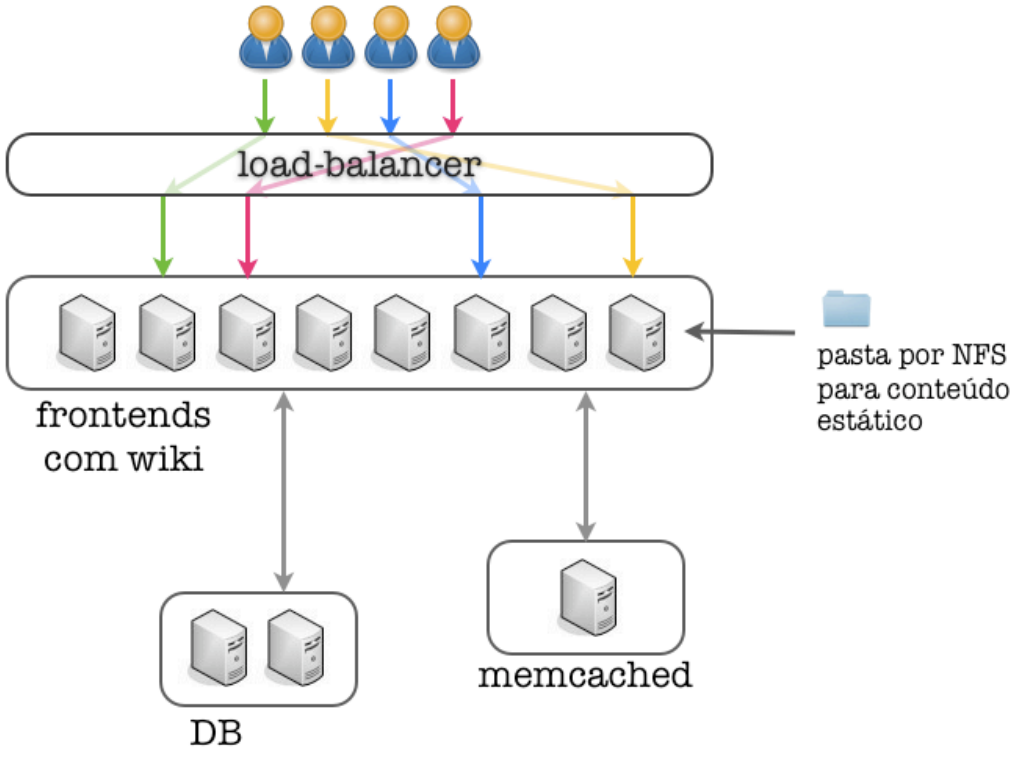

Figura 11 – Arquitectura de rede utilizada para alojar a Wiki.

3.3.4 Testes

Os testes realizados à plataforma serão na sua totalidade com o objectivo de assegurar um desempenho satisfatório num cenário de imensos pedidos. Fizemos os esforços para garantir que temos uma plataforma que não necessita de computar a página inicial cada vez que esta é pedida, visto que usa sistemas de memória temporária (cache).

Este sistema de *cache*, o **Memcached**, opera armazenado objectos na memória duma máquina dedicada, para evitar os tempos de acesso ao armazenamento secundário (disco rígido), que por mais eficiente que seja, nunca conseguirá as velocidades obtidas na utilização de memória principal (RAM).
Para além disso, todo o conteúdo estático será servido dum domínio diferente para que se possa utilizar definições diferentes do que os utilizados para servir conteúdo dinâmico. Desta forma também se evita que os pedidos de conteúdo estático incluam informações desnecessárias como é o caso dos *cookies* – são pedaços de informação relevante à aplicação e a outras utilizadas no mesmo domínio – que apenas irão aumentar o volume de dados trocados entre o browser do utilizador e o servidor.

O conteúdo será também servido com cabeçalhos HTTP que reforcem a *cache* efectuada a nível dos *browsers.*

Não é possível apresentar dados concretos dos testes finais uma vez que a plataforma ainda está à espera de ser movida para o ambiente de produção. Por motivos alheios ao desenvolvimento da plataforma, como já foi referido, houve um atraso no lançamento da mesma (ver capítulo 2.2 deste relatório).

4. Conclusões

Após conclusão do projecto, o facto de termos partido de soluções **open source** facilitou o arranque do projecto, limitado-nos a complementar o conjunto de funcionalidades já oferecido pelas duas plataformas.

Isto permitiu-nos adoptar uma metodologia de desenvolvimento baseado em iterações, permitindo readaptar o percurso do projecto de acordo com o *feedback* fornecido pela equipa de testes, coisa que seria difícil num processo como o Cascata.

Para além disso, o facto de estarmos em permanente contacto com a comunidade do **eTicket** deu-nos algumas vantagens visto que não só ajudámos o projecto com funcionalidades novas, como ajudámos a levá-lo para a frente com o nosso *feedback* e com as discussões que começámos no seio da comunidade.

As wikis, são de facto ferramentas muito poderosas, no entanto, senti que o uso duma wiki para este efeito prendia-se mais com o facto de permitir controlo de versões das páginas e objectos multimédia do que pelo factor "editável por todos" inerente e próprio das wikis.

No geral, o projecto verificou-se bastante estimulante uma vez que lidou com aspectos importantes no ecossistema do desenvolvimento de aplicações para a web. Utilizou o universo **open source** para acrescentar valor a um projecto já existente, o que aumenta o impacto do trabalho levado a cabo no âmbito deste projecto.

Para além disso, focou uma área que me interessa bastante e que ganha cada vez mais relevância no desenvolvimento de aplicações no mundo real; o da usabilidade. De que serve desenvolver uma aplicação complexa se provoca frustração naqueles que a usam?

Para finalizar, as preocupações em termos do desempenho e da arquitectura de sistemas foram questões que me ajudaram a consolidar conceitos já abordados nalgumas cadeiras do mestrado de Engenharia Informática.

Ainda assim, acredito que há espaço para melhorar ambas as plataformas. Há vários aspectos na plataforma de atendimento de pedidos de suporte que beneficiariam de testes formais de usabilidade para garantir que o nível de satisfação dos que a usam continua alto.

Na wiki, apesar do trabalho em falta ser maioritariamente de edição de conteúdos, há mecanismos que podem ser apurados no que toca à edição textual. Os operadores continuam a ter que apreender conceitos de **Wikitext**, o que decorreu sem problemas. No entanto, há pessoas com mais dificuldade em utilizarem linguagens deste tipo. Um editor visual, com ajudas de edição rápida, seria bastante vantajoso para algumas pessoas. Apesar de termos consultado algumas soluções aplicáveis a esta plataforma, nenhuma verificava um nível de maturidade suficiente que justificasse a inclusão da mesma no sistema. Ficou, portanto, para trabalho futuro.

5. Referências

[1] SourceForge, http://www.sourceforge.net, em Julho de 2007

[2] FreshMeat, http://www.freshmeat.com, em Julho de 2007

[3] OTRS, http://otrs.org/, em Julho de 2007

[4] osTicket, http://www.osticket.com/, em Julho de 2007

[5] eTicket, http://eticket.sourceforge.net, desde Julho de 2007 a Maio de 2008.

[6] WikiMatrix, http://www.wikimatrix.org, em Julho de 2007

[7] MediaWiki, http://www.mediawiki.org, desde Julho de 2007 a Maio de 2008

[8] Wikipedia, http://wikipedia.org

[9] Memcached, http://www.danga.com/memcached/

[10] Oito Regras de Ouro de Ben Shneiderman, http://en.wikipedia.org/wiki/Shneiderman%27s_rules_for_design

[11] Jeremy Keith, DOM Scripting. Friends of ED, 2005.

[12] John Allsopp, Microformats: Empowering your markup for Web 2.0. Friends of ED, 2007.